

A Lasserre-based $(1 + \varepsilon)$ -approximation for $Pm \mid p_j = 1, \text{prec} \mid C_{\max}$

Elaine Levey* and Thomas Rothvoss†

University of Washington, Seattle

September 28, 2015

Abstract

In a classical problem in scheduling, one has n unit size jobs with a *precedence order* and the goal is to find a schedule of those jobs on m identical machines as to minimize the makespan. It is one of the remaining four open problems from the book of Garey & Johnson whether or not this problem is **NP**-hard for $m = 3$.

We prove that for any fixed ε and m , a Sherali-Adams / Lasserre lift of the time-index LP with a slightly super poly-logarithmic number of $r = (\log(n))^{\Theta(\log \log n)}$ rounds provides a $(1 + \varepsilon)$ -approximation. This implies an algorithm that yields a $(1 + \varepsilon)$ -approximation in time $n^{O(r)}$. The previously best approximation algorithms guarantee a $2 - \frac{7}{3m+1}$ -approximation in polynomial time for $m \geq 4$ and $\frac{4}{3}$ for $m = 3$. Our algorithm is based on a recursive scheduling approach where in each step we reduce the correlation in form of long chains. Our method adds to the rather short list of examples where hierarchies are actually useful to obtain better approximation algorithms.

1 Introduction

One of the landmarks in the theory of scheduling is the paper of Graham [Gra66] from 1966, dealing with the following problem: suppose we have a set J of n jobs, each one with a running time p_j along with m identical parallel machines that we can use to process the jobs. Moreover, the input contains a *precedence order* on the jobs; we write $j < j'$ if job j has to be completed before job j' can be started. The goal is to schedule the jobs in a non-preemptive fashion so that the *makespan* is minimized. Here, the

*Email: elevey@cs.washington.edu. Supported by NSF grant CCF 1420381.

†Email: rothvoss@uw.edu. Supported by NSF grant 1420180 with title “*Limitations of convex relaxations in combinatorial optimization*” and a Sloan Research Fellowship. File compiled on September 28, 2015, 00:25.

makespan gives the time that the last job is finished. In the *3-field notation*¹, this problem is abbreviated as $P \mid \text{prec} \mid C_{\max}$. Graham showed that the following *list schedule* gives a $(2 - \frac{1}{m})$ -approximation on the makespan: compute an arbitrary topological ordering of the jobs and whenever a machine becomes idle, select the first available job from the list. It had been known since the late 70's that it is **NP**-hard to approximate the problem better than within a factor of $4/3$ due to Lenstra and Rinnooy Kan [LK78] and Schuurman and Woeginger [SW99] prominently placed the quest for any improvement on their well known list of 10 open problems in scheduling. Finally in 2010, Svensson [Sve10] showed that assuming a variant of the unique games conjecture [BK09], there is no $(2 - \epsilon)$ -approximation algorithm for $P \mid \text{prec}, p_j = 1 \mid C_{\max}$. However, for unit size jobs, Lam and Sethi [LS77] analyzed an algorithm of Coffman and Graham and showed that it provides a slightly better guarantee of $2 - \frac{2}{m}$ for $P \mid \text{prec}, p_j = 1 \mid C_{\max}$. Later, Gangal and Ranade [GR08] gave an algorithm with a $2 - \frac{7}{3m+1}$ guarantee for $m \geq 4$.

In a typical scheduling application, the number of jobs might be huge compared to the number of machines, which does justify to ask for the complexity status of such problems if the number m of machines is a constant. Even under the additional restriction of unit size jobs, no better approximation result is known. In fact, it is one of the remaining four open problems from the book of Garey and Johnson [GJ79] whether $P3 \mid \text{prec}, p_j = 1 \mid C_{\max}$ is even **NP**-hard. Also Schuurman and Woeginger [SW99] list under “Open Problem 1” the question whether there is a PTAS for this problem (recall that for $m = 2$, the result of [LS77] gives an optimum schedule).

To understand where the lack of progress is coming from, one has to go back to the list scheduling algorithm of Graham. If we schedule the jobs in a greedy manner, then one can argue that there is always a *chain* of jobs $j_1 < j_2 < \dots < j_k$ so that at any point in time either all m machines were fully busy or a job from that chain was processed. Since both quantities, the *load* $\frac{1}{m} \sum_{j \in J} p_j$ and the length of any chain are lower bounds on any schedule, we can conclude that the schedule has length at most $2 \cdot \text{OPT}$. One can shave off a factor of $\frac{1}{m}$ even for general running times, by observing that the processing times of the jobs in the longest chain do not need to be again counted in the load bound. Also the papers [LS77] and [GR08] effectively rely on those two lower bounds. In fact, [Cha95] showed that a large class of algorithms including the ones of [Gra66, GR08] cannot beat a bound of $2 - \frac{2}{\sqrt{m}}$; moreover Graham's algorithm is indeed not better than a $(2 - \frac{2}{m})$ -approximation for unit size jobs, see [GR08].

Of course, one always has the option to study the strength of linear programs for an optimization problem. The natural one for $Pm \mid \text{prec}, p_j = 1 \mid C_{\max}$ is certainly the following *time-indexed LP*: For a parameter T that denotes the length of the time horizon,

¹In the 3-field notation, the first field specifies the available processors, the 2nd field the jobs and the last field the objective function. In our case, Pm means that we have m identical machines; $p_j = 1$, prec indicates that the jobs have unit length and precedence constraints and the last field C_{\max} specifies that the objective function is to minimize the maximum completion time.

we define a set $K(T)$ as the set of fractional solutions to:

$$\begin{aligned}
\sum_{t=1}^T x_{j,t} &= 1 \quad \forall j \in J \\
\sum_{j \in J} x_{j,t} &\leq m \quad \forall t \in [T] \\
\sum_{t' \leq t} x_{i,t'} &\geq \sum_{t' \leq t+1} x_{j,t'} \quad \forall i < j \quad \forall t \in [T] \\
x_{j,I} &= \sum_{t \in I} x_{j,t} \quad \forall j \in J \quad \forall \text{ intervals } I \text{ of } T \\
0 \leq x_{j,t} &\leq 1 \quad \forall j \in J \quad \forall t \in [T]
\end{aligned} \tag{1}$$

Here $x_{j,t}$ is a decision variable that is supposed to tell whether job $j \in J$ is scheduled in time slot $t \in \{1, \dots, T\}$. The constraints guarantee that in an integral solution each job is assigned to one time slot; no time slot receives more than m jobs and for a pair of jobs $i < j$, job i has to be scheduled before j . Note that we have introduced redundant looking variables $x_{j,I}$ which for each interval $I = \{a, \dots, b\}$ with $1 \leq a \leq b \leq T$ denote whether job j will be scheduled somewhere within that interval I . Those variables obviously do not change the feasibility of the LP, but they will be very helpful later for notational purposes.

Unsurprisingly, this LP has a constant integrality gap as one can see from the following construction: take k blocks J_1, \dots, J_k of $|J_i| = m+1$ jobs each and define the precedence order so that all the jobs in J_i have to be finished before any job in J_{i+1} can be started. Any integral schedule needs two time units per block, hence $OPT = 2k$. On the other hand, the LP solution can schedule the $m+1$ jobs of each block “in parallel”, each at a rate of $\frac{m}{m+1}$ and finish the schedule after $k \cdot \frac{m+1}{m}$ time units in which each machine has always been fully busy. This results in an integrality gap of at least $2 - \frac{2}{m+1}$.

It has been long known, that in principle one can take the linear program for any optimization problem and strengthen it automatically by applying an LP or SDP hierarchy lift. We will provide formal definitions later, but basically these operators ensure that for any set of at most r variables, the LP solution indeed lies the convex hull of integral combinations. Here, r is the number of *levels* or *rounds* and one typically needs time $n^{O(r)}$ to solve an r -level hierarchy.

Some known approximation results have been reinterpreted in hind side in this framework, for example a constant number of Lasserre rounds applied to a basic LP suffice for the Goemans-Williamson algorithm for MaxCut [GW95] and also a constant number of Lasserre rounds imply the triangle inequalities in the $O(\sqrt{\log n})$ -approximation algorithm by Arora, Rao and Vazirani [ARV09]. Moreover, the subspace enumeration component in the subexponential time algorithm of Arora, Barak and Steurer [ABS10] for Unique Games could be replaced with a Lasserre SDP. However, there are relatively few results where hierarchies have been genuinely useful (at least fewer than researchers have hoped for). For example Chlamtác [Chl07] used SDP hierarchies to find better colorings in 3-colorable graphs and Raghavendra and Tan [RT12] apply them to obtain approximation algorithms for CSPs with cardinality constraints. An application to

color hypergraphs can be found in [CS08]. Hierarchies also turned out to be the right approach for Sparsest Cut in bounded tree width graphs, see the paper by Chlamtáč, Krauthgamer and Raghavendra [CKR10] and the 2-approximation by Gupta, Talwar and Witmer [GTW13].

1.1 Our contribution

Our main result is that a Lasserre lift with $(\log(n))^{O((m^2/\varepsilon^2) \cdot \log \log n)}$ rounds closes the integrality gap of the above LP to at most $1 + \varepsilon$. This implies:

Theorem 1. *For the problem $Pm \mid \text{prec}, p_j = 1 \mid C_{\max}$ one can compute a $(1 + \varepsilon)$ -approximate solution in time $n^{O(r)}$ where $r := (\log(n))^{O((m^2/\varepsilon^2) \cdot \log \log n)}$.*

This gives a partial answer to one of the question under “Open Problem 1” in [SW99] which asked whether there is a PTAS for this problem. In a Dagstuhl workshop, Mathieu [Dag10] asked the more specific question whether the Sherali-Adams hierarchy gives a $(1 + \varepsilon)$ -approximation after $c(\varepsilon, m)$ rounds. We also make progress on the question from the book of Garey and Johnson [GJ79] by improving the $\frac{4}{3}$ -polynomial time approximation for $m = 3$ [LS77] to a $1 + \varepsilon$ in slightly more than quasi-polynomial time. In particular, this implies that $Pm \mid \text{prec}, p_j = 1 \mid C_{\max}$ is not **APX**-hard, assuming that $\mathbf{NP} \not\subseteq \mathbf{DTIME}(n^{\log(n)^{O(\log \log n)}})$.

We should point out that the same result could be proven with other hierarchies such as Sherali-Adams. We choose to work with the Lasserre hierarchy as it allows a cleaner mathematical definition, in our opinion.

2 The Lasserre hierarchy

In this section, we want to discuss the *Lasserre hierarchy* which was introduced in [Las01a, Las01b]. For the moment, let $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ be some linear relaxation of an arbitrary binary optimization problem. In any words, K is a relaxation of the integral hull $K_I := \text{conv}(K \cap \{0, 1\}^n)$. Let us make the observation that for a vector $x \in K_I$ there is a distribution X over $K \cap \{0, 1\}^n$ so that x is the vector of *marginal probabilities*, that means $\Pr[X_i = 1] = x_i$. If we only have $x \in K$, then in general, there is no such probability distribution. We can also make the observation that the vector x contains no information on the probability of *joint events* such as $\Pr[X_i = X_j = 1]$. The idea behind LP or SDP hierarchies in general, is to add additional variables to the system that contain this extra information of probabilities for joint events.

In case of Lasserre’s hierarchy, we consider a vector $y \in \mathbb{R}^{2^{[n]}}$ which is a 2^n -dimensional vector, indexed by subsets of variables. The motivation is that for any subset $I \subseteq \{1, \dots, n\}$ of variable indices, we would want the entry y_I to be the probability $y_I = \Pr[\bigwedge_{i \in I} (X_i = 1)]$. In particular this means that $y_{\{i\}} = x_i$ and $y_{\emptyset} = 1$. Note that any probability distribution over $\{0, 1\}^n$ is determined by $2^n - 1$ many values and one can use the *Inclusion-Exclusion*

Formula to derive all probabilities from just knowing the y_I 's. Moreover, one can write down explicitly the constraints that ensure that the vector y encodes a probability distribution. However, as y is 2^n -dimensional and we aim at (quasi-)polynomial time algorithms, we will only introduce constraints that involve entries y_I for $|I| \leq 2r + 1$ and r is the *number of rounds* or the *level* of our solution. This way we can still solve the system in time $n^{O(r)}$.

For a symmetric matrix M , we will write $M \geq 0$ if M is *positive semidefinite*, that means if all its Eigenvalues are non-negative.

Definition. Let $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. We define the r -th level of the Lasserre hierarchy $\text{LAS}_r(K)$ as the set of vectors $y \in \mathbb{R}^{2^{[n]}}$ that satisfy

$$M_r(y) := (y_{I \cup J})_{|I|, |J| \leq r} \geq 0; \quad M_r^\ell(y) := \left(\sum_{i=1}^n A_{\ell i} y_{I \cup J \cup \{i\}} - b_\ell y_{I \cup J} \right)_{|I|, |J| \leq r} \geq 0 \quad \forall \ell \in [m]; \quad y_\emptyset = 1.$$

The matrix $M_r(y)$ is called the *moment matrix* of y and the second type $M_r^\ell(y)$ is called *moment matrix of slacks*. Furthermore, let $\text{LAS}_r^{\text{proj}}(K) := \{(y_{\{1\}}, \dots, y_{\{n\}}) \mid y \in \text{LAS}_r(K)\}$ be the *projection on the original variables*.

We want to emphasize that the r -th level of Lasserre cares only for entries y_I with $|I| \leq 2r + 1$ – all other entries are ignored (if the reader prefers a proper definition, he might impose $y_I = 0$ for $|I| > 2r + 1$). The set of positive semidefinite matrices forms a non-polyhedral convex cone, thus also $\text{LAS}_r(K)$ is a convex set. The separation problem for $\text{LAS}_r(K)$ boils down to computing a negative Eigenvector of the involved matrices, which can be done in polynomial time. As a consequence, if m denotes the number of linear constraints, one can optimize over $\text{LAS}_r(K)$ in time $n^{O(r)} \cdot m^{O(1)}$ up to numerical errors that can be neglected.

Intuitively speaking, the constraint $M_r(y) \geq 0$ takes care that the variables are consistent (e.g. it guarantees that $y_{\{1,2\}} \in [y_{\{1\}} + y_{\{2\}} - 1, \min\{y_{\{1\}}, y_{\{2\}}\}]$). Moreover, the psdness of the ℓ th slack matrices $M_r^\ell(y)$ guarantees that y satisfies the ℓ -th linear constraint.

Let us begin with discussing some of the basic properties, especially that $\text{LAS}_r(K)$ is indeed a relaxation of $K \cap \{0, 1\}^n$ and that any set of at most r variables looks like the convex combination of integral solutions. For a proof, we refer to the survey of Laurent [Lau03] or the lecture notes of the 2nd author [Rot13].

Lemma 2. Let $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ and $y \in \text{LAS}_r(K)$ for $r \geq 0$. Then the following holds:

- a) $K \cap \{0, 1\}^n = \text{LAS}_n^{\text{proj}}(K) \subseteq \dots \subseteq \text{LAS}_1^{\text{proj}}(K) \subseteq \text{LAS}_0^{\text{proj}}(K) \subseteq K$.
- b) One has $0 \leq y_J \leq y_I \leq 1$ for all $I \subseteq J$ with $0 \leq |I| \leq |J| \leq r$.
- c) Suppose $r \geq 1$ and $i \in [n]$ is a variable with $0 < y_i < 1$. If we define

$$z_I^{(1)} := \frac{y_{I \cup \{i\}}}{y_i} \quad \text{and} \quad z_I^{(0)} := \frac{y_I - y_{I \cup \{i\}}}{1 - y_i}$$

then we have $y = y_i \cdot z^{(1)} + (1 - y_i) \cdot z^{(0)}$ with $z^{(0)}, z^{(1)} \in \text{LAS}_{r-1}(K)$ where $z_i^{(0)} = 0$, and $z_i^{(1)} = 1$.

If we have vector $y \in \text{LAS}_r(K)$ and a variable i with $0 < y_i < 1$, then *conditioning on* $x_i = 1$ means to replace the vector y with the vector $z^{(1)} \in \text{LAS}_{r-1}(K)$ with $z_i^{(1)} = 1$. On the other hand, if we are *conditioning on* $x_i = 0$, then it means to replace y by $z^{(0)} \in \text{LAS}_{r-1}(K)$ with $z_i^{(0)} = 0$. The correctness for this procedure follows right from property c).

2.1 The Lasserre Lift for the scheduling LP

Now, let us go back to the scheduling LP and discuss the properties of the Lasserre lift, applied to $K(T)$ from Eq. (1). For a vector $y \in \text{LAS}_r(K(T))$ and a job j , we define $\text{supp}(j, y) := \{t \in [T] \mid y_{j,t} > 0\}$ as the *support* of the job. Here we should mention that for the sake of a simpler notation we address the singleton variables $y_{\{(j,t)\}}$ as $y_{j,t}$ whenever convenient.

Lemma 3. *Let $y \in \text{LAS}_r(K(T))$. Then the following holds:*

- a) *If $\text{supp}(j, y) \subseteq \{1, \dots, t_0\}$ and $\text{supp}(j', y) \subseteq \{t_0, \dots, T\}$ then it is impossible that $j' < j$.*
- b) *Suppose that y' is derived from y by any sequence of $k \leq r$ conditionings. Then $y' \in \text{LAS}_{r-k}(K(T))$ and $\text{supp}(j, y') \subseteq \text{supp}(j, y)$ for all jobs $j \in J$.*

Proof. Claim a) already follows from the fact that y represents a feasible fractional solution to $K(T)$. For claim b), by induction it suffices to consider the case where y' is derived by a single conditioning. Let's say we conditioned on $x_{j_0, t_0} = 1$ and suppose that $y_{\{(j_1, t_1)\}} = 0$. Then from the definition in Lemma 2.c we know that

$$y'_{\{(j_1, t_1)\}} = \frac{y_{\{(j_0, t_0), (j_1, t_1)\}}}{y_{\{(j_0, t_0)\}}} \leq \frac{y_{\{(j_1, t_1)\}}}{y_{\{(j_0, t_0)\}}} = 0,$$

using Lemma 2.b in the inequality. □

3 An overview

In this section, we will give an overview over the different steps in our algorithm; the detailed implementation of some of the steps will be given in the following Sections 4, 5 and 6. For a given time horizon T , a *feasible schedule* is an assignment $\sigma : J \rightarrow \{1, \dots, T\}$ with $|\sigma^{-1}(t)| \leq m$ for all $t \in [T]$ and for all $j, j' \in J$ one has $j < j' \Rightarrow \sigma(j) < \sigma(j')$.

Formally, our main technical theorem is as follows:

Theorem 4. *For any Lasserre solution $y \in \text{LAS}_r(K(T))$ with $r := (\log n)^{O((m^2/\varepsilon^2) \cdot \log \log n)}$, one can find a feasible schedule $\sigma : J \rightarrow \mathbb{N}$ of the jobs in time $n^{O(r)}$ so that $\max_{j \in J} \sigma(j) \leq (1 + \varepsilon) \cdot T$.*

To obtain a $(1+\epsilon)$ -approximation, we can find the minimum value of T so that $\text{LAS}_r(K(T)) \neq \emptyset$ with binary search and then compute a vector $y \in \text{LAS}_r(K(T))$. In particular, by virtue of being a relaxation, that value of T will satisfy $T \leq \text{OPT}$, where OPT is the makespan of the optimum schedule. For the sake of a simpler notation, we will assume that T is a power of 2 — if $2^{z-1} < T \leq 2^z$ for some integer z , then one can add $m \cdot (2^z - T)$ many dummy jobs that all depend on each original job so that the algorithm will schedule the dummy jobs at the very end. Moreover we will assume that $\frac{1}{\epsilon}, m \leq \log_2(n)$ as otherwise the bound is meaningless.

The main routine of our algorithm will schedule jobs only within the time horizon T of the Lasserre solution, but we will allow it to *discard* jobs. Formally this means, we will find an assignment $\sigma : J \setminus J_{\text{discarded}} \rightarrow [T]$ that will not have assigned slots to jobs in $J_{\text{discarded}}$. Such an assignment will still be called “feasible” if apart from the load bound, the condition $j < j' \Rightarrow \sigma(j) < \sigma(j')$ is satisfied for all $j, j' \in J \setminus J_{\text{discarded}}$. In particular dependencies with discarded jobs play no role in this definition.

The reason for this definition is that one can easily insert the discarded jobs at the very end of the algorithm:

Lemma 5. *Any feasible schedule $\sigma : J \setminus J_{\text{discarded}} \rightarrow \{1, \dots, T\}$ can be modified in polynomial time to a feasible schedule $\sigma^* : J \rightarrow \{1, \dots, T + |J_{\text{discarded}}|\}$ (which also includes the previously discarded jobs).*

Proof. Select any job $j^* \in J_{\text{discarded}}$. Since σ is a valid schedule which respects all precedence constraints in $J \setminus J_{\text{discarded}}$, there must be a time t^* so that all jobs $j < j^*$ have $\sigma(j) \leq t^*$ and all jobs j with $j^* < j$ have $\sigma(j) > t^*$. Then we insert an extra time unit after time t^* ; in this extra time slot, we only process j^* . Then we continue the procedure with inserting the next job from $J_{\text{discarded}} \setminus \{j^*\}$. \square

Now, let us introduce some notation: We can imagine the precedence order “ $<$ ” as a directed transitive graph $G = (J, E)$ with the nodes as jobs and edges $(j, j') \in E \Leftrightarrow j < j'$. In that view, let $\delta^+(j) := \{j' \in J \mid j < j'\}$ be the jobs depending on j and let $\delta^-(j) := \{j' \in J \mid j' < j\}$ be the jobs on which j depends. Note that $\delta^+(j)$ and $\delta^-(j)$ are always distinct. We abbreviate $\delta(j) := \delta^+(j) \cup \delta^-(j)$ as the jobs that have any dependency with j . Finally, for a subset of jobs $J' \subseteq J$, let $\Delta(J') := \max\{|\delta(j) \cap J'| + 1 \mid j \in J'\}$ be the maximum degree of a node in the subgraph induced by J' , counting also the node himself.

We partition the time horizon $[T]$ into a balanced binary family \mathcal{I} of intervals of lengths $T, \frac{T}{2}, \frac{T}{2^2}, \dots, 2, 1$. Let $\mathcal{I} := \mathcal{I}_0 \dot{\cup} \dots \dot{\cup} \mathcal{I}_L$ be the *binary laminar family of intervals* that we obtain by repeatedly partitioning intervals into two equally-sized subintervals. Recall that each *level* \mathcal{I}_ℓ contains 2^ℓ many interval $I \in \mathcal{I}_\ell$; each one consisting of $|I| = \frac{T}{2^\ell}$ many time units.

Our algorithm will schedule the jobs in a recursive manner. The main claim is that for any interval I^* , Lasserre solution y^* and a set of jobs J^* with $\text{supp}(j, y^*) \subseteq I^*$, we can schedule almost all jobs from J^* within I^* while respecting all precedence constraints.

We use parameters $k := \frac{c_1 m}{\epsilon} \log \log(T)$ where $c_1 > 0$ is a large enough constant that

we will choose in Section 6, and $\delta := \frac{\varepsilon}{8k^2m2^{k^2}\log(T)}$. To get some intuition for the parameters, considering ε and m as fixed constants, one would have $k = \Theta(\log \log n)$ and $\delta = 1/\log(n)^{\Theta(\log \log n)}$. Formally, the main technical lemma is the following:

Lemma 6. *Fix $\varepsilon > 0$. Let $I^* \in \mathcal{I}$ be an interval from the balanced family. Let $y^* \in \text{LAS}_{r^*}(K(T))$ be a Lasserre solution with*

$$r^* \geq \log_2(T^*) \cdot 2mk^2 \cdot 2^{k^2} / \delta.$$

Let $J^ \subseteq \{j \in J : y_{j,I^*}^* = 1\}$ be some jobs fully assigned to I^* . Then one can find a feasible assignment $\sigma : J^* \setminus J_{\text{discarded}}^* \rightarrow I^*$ that discards only*

$$|J_{\text{discarded}}^*| \leq \frac{\varepsilon}{2} \cdot \frac{\log_2(|I^*|)}{\log_2(T)} \cdot |I^*| + \frac{\varepsilon}{2m} \cdot |J^*|$$

many jobs.

Before we move on to explain the procedure behind Lemma 6, we want to argue that it implies our main result:

Proof of Theorem 4. We set $I^* := \{1, \dots, T\}$ and $y^* := y$, then $J^* := J$ is a valid choice as trivially $y_{j,\{1,\dots,T\}} = 1$ for any job. To satisfy the requirement of Lemma 6 we need

$$\log_2(T) \cdot \frac{2mk^2 \cdot 2^{k^2}}{\delta} \leq (\log(n))^{O((\frac{m}{\varepsilon})^2 \log_2 \log_2(n))}$$

many levels of the hierarchy. Here we use that $k = O(\frac{m}{\varepsilon} \log \log T)$, hence $2^k = (\log(T))^{\Theta(m/\varepsilon)}$ and $2^{k^2} = (\log(T))^{O((\frac{m}{\varepsilon})^2 \log \log T)}$ (we want to point out that many of the lower order terms are absorbed into the O -notation of the exponent and we assume that $\frac{1}{\varepsilon}, m \leq \log_2(n)$). Then Lemma 6 returns a valid assignment $\sigma : J \setminus J_{\text{discarded}} \rightarrow [T]$ that discards only

$$|J_{\text{discarded}}| \leq \frac{\varepsilon}{2} \cdot \frac{\log_2(T)}{\log_2(T)} \cdot T + \frac{\varepsilon}{2m} \cdot |J| \leq \varepsilon \cdot T$$

many jobs. Inserting those discarded jobs via Lemma 5 then results in a feasible schedule of makespan at most $(1 + \varepsilon) \cdot T$. \square

The rest of the manuscript will be devoted to proving Lemma 6. We fix a constant $\varepsilon > 0$ as the target value for our approximation ratio and denote $T^* := |I^*|$ as the length of our interval.

Let us first argue, how to handle the base case, which for us is if $\log_2(T^*) \leq k^2$. In that case, we have at most $mT^* \leq m2^{k^2}$ jobs. Hence, the Lasserre lift has $r^* \geq mT^*$ many levels and one can repeatedly condition on events $x_{j,t} = 1$ for $j \in J^*$ and $t \in I^*$ until one arrives at a Lasserre vector y^{**} with $y_{j,t}^{**} \in \{0, 1\}$ for all $j \in J^*$. This then represents a valid schedule of jobs J^* in the interval I^* without that any jobs need to be discarded.

We now come to a high level description of the algorithm. Let $\mathcal{I}_0^*, \dots, \mathcal{I}_{\log_2(T^*)}^*$ be the family of subintervals of I^* , where \mathcal{I}_ℓ^* contains 2^ℓ intervals of length $\frac{T^*}{2^\ell}$ each (see Figure 1). For a job $j \in J^*$, we define $\ell(j, y^*) := \max\{\ell : \exists I \in \mathcal{I}_\ell^* \text{ with } y_{j,I}^* = 1\}$ as the level that *owns* the job in the current Lasserre solution. We also abbreviate $J(\ell, y^*) := \{j \in J^* \mid \ell(j, y^*) = \ell\}$ as all jobs owned by level ℓ . The algorithm is as follows:

- **Step 1:** Starting with the Lasserre solution y^* , we can iteratively condition on events until we arrive at a Lasserre vector y^{**} that has the property that for any interval $I \in \mathcal{I}_0^* \cup \dots \cup \mathcal{I}_{k^2-1}^*$, the jobs owned by that interval have small dependence degree, that means $\Delta(J(I, y^{**})) \leq \delta|I|$. If we then consider the set of jobs $J^{**} := \{j \in J^* \mid 0 \leq \ell(j, y^{**}) < k^2\}$ owned by the first k^2 levels, the longest chain in J^{**} will contain at most $k^2 \delta T^*$ jobs. We will show in Section 4 that the number of conditionings that are needed, can be upperbounded by $2mk^2 \cdot 2^{k^2}/\delta$, which implies that $y^{**} \in \text{LAS}_{r^* - 2mk^2 \cdot 2^{k^2}/\delta}(K(T))$.
- **Step 2:** From now on, we work with the modified Lasserre solution y^{**} . We select a level index $\ell^* \in \{k, \dots, k^2\}$ and partition the jobs in J^* in three different groups:
 - The jobs on the top levels: $J_{\text{top}} := J(0, y^{**}) \cup \dots \cup J(\ell^* - k - 1, y^{**})$
 - The jobs on the k middle levels: $J_{\text{middle}} := J(\ell^* - k, y^{**}) \cup \dots \cup J(\ell^* - 1, y^{**})$
 - The jobs on the bottom levels: $J_{\text{bottom}} := J(\ell^*, y^{**}) \cup \dots \cup J(\log_2(T^*), y^{**})$

Then we discard all jobs in J_{middle} . In Section 6 we will describe how the index ℓ^* is chosen and in particular we will provide an upper bound on the number of discarded middle jobs.

- **Step 3:** In this step, we will find a schedule for the bottom jobs. For this purpose, we call Lemma 6 recursively for each interval $I \in \mathcal{I}_{\ell^*}^*$ with a copy of the Lasserre vector y^{**} and jobs $J_I := \{j \in J_{\text{bottom}} \mid y_{j,I}^{**} = 1\}$. Then Lemma 6 returns a valid schedule of the form $\sigma_I : J_I \setminus J_{I,\text{discarded}} \rightarrow I$ for each interval $I \in \mathcal{I}_{\ell^*}^*$. Let $J_{\text{bottom-discarded}} := \bigcup_{I \in \mathcal{I}_{\ell^*}^*} J_{I,\text{discarded}} \subseteq J_{\text{bottom}}$ be the union of jobs that were discarded in those calls. The partial schedules σ_I satisfy $|\sigma_I(t)| \leq m$ for $t \in I$ and $|\sigma_I^{-1}(t)| = 0$ for $t \notin I$. We combine those schedules to a schedule $\sigma : J_{\text{bottom}} \setminus J_{\text{bottom-discarded}} \rightarrow I^*$. From the disjointness of the intervals, it is clear that again $|\sigma^{-1}(t)| \leq m$ for all $t \in I^*$. Moreover, if $j < j'$ and $j, j' \in J_I$ for some interval $I \in \mathcal{I}_{\ell^*}^*$, then by the inductive hypothesis $\sigma(j) < \sigma(j')$. On the other hand, if $j \in J_I$ and $j' \in J_{I'}$ then we know by Lemma 3.b that I had to come before I' since $y_{j,I}^{**} = 1 = y_{j',I'}^{**}$.
- **Step 4:** We continue working with the previously constructed schedule σ that schedules the non-discarded bottom jobs. In this step, we will extend the schedule σ and insert the jobs of J_{top} in the remaining free slots. We will prove in Section 5 that this can be done without changing the position of any scheduled bottom job and without violating any precedence constraints. Again, we allow that the procedure discards a small number of additional jobs from J_{top} that we will account for later. Eventually, the schedule σ satisfies the claim for Lemma 6.

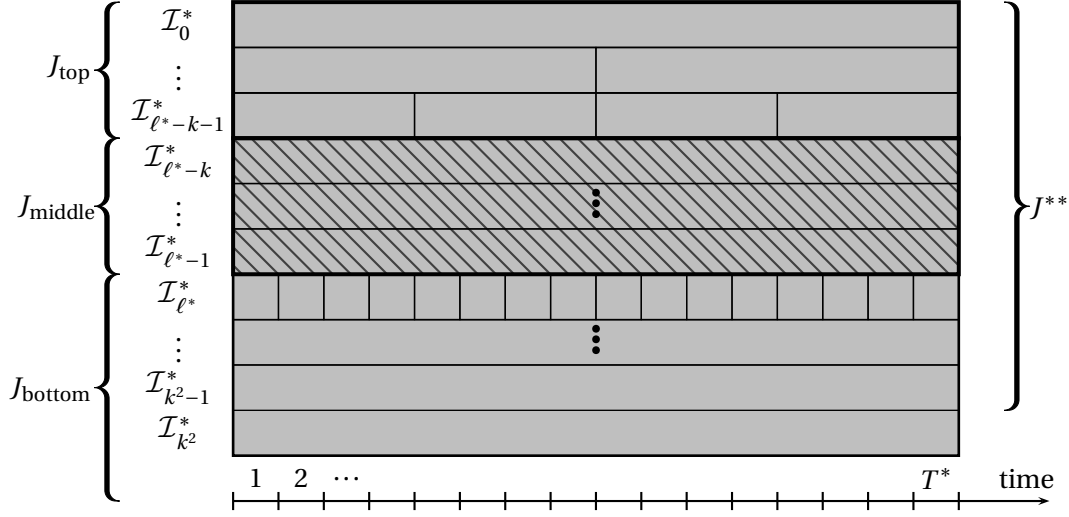


Figure 1: Binary dissection of the interval I^* used in the algorithm behind Lemma 6.

The intuition behind the algorithm is as follows: When we call the procedure recursively for intervals $I \in \mathcal{I}_{\ell^*}^*$ we cannot control where the jobs J_I will be scheduled within that interval I . In particular the decisions made in different intervals $I, I' \in \mathcal{I}_{\ell^*}^*$ will in general not be consistent. But the discarding of the middle jobs has created a gap between the top jobs and the bottom jobs in the sense that the intervals of the top jobs are at least a factor 2^k longer than intervals of the bottom jobs. For a top job $j \in J_{\text{top}}$ we will be pessimistic and assume that all the bottom jobs that j depends on will be scheduled just at the very end of their interval. Still, as those intervals are very short, we will be able to argue that the loss in the flexibility is limited and most of the top jobs can be processed. As a second crucial ingredient, the conditioning had the implication that the top jobs do not contain any long chains any more. This will imply that a greedy schedule of the top jobs will leave little idle time, resulting in only few discarded top jobs.

4 Step (1) — Reducing dependence

In this section we will implement “Step (1)” and show that we can reduce the maximum dependence degrees of the jobs owned by the first k^2 levels in order to bound the length of chains. We will do this by conditioning on up to $2mk^2 \cdot 2^{k^2}/\delta$ many variables. Recall that we are considering an interval I^* and a subset of jobs $J^* \subseteq J$ that the current Lasserre solution y^* fully schedules within I^* . For one of the subintervals $I \in \mathcal{I}_{\ell}^*$ below I^* , let us abbreviate $J(I, y^*) := \{j \in J(\ell, y^*) \mid y_{j,I}^* = 1\}$ as the jobs owned by that particular interval.

Lemma 7. *Let $y^* \in \text{LAS}_{r^*}(K(T))$. Then one can find an induced solution $y^{**} \in \text{LAS}_{r^{**}}(K(T))$ with $r^{**} := r^* - 2mk^2 \cdot 2^{k^2}/\delta$ so that $\Delta(J(I, y^{**})) \leq \delta \cdot |I|$ for all intervals $I \in \mathcal{I}_0^* \cup \dots \cup \mathcal{I}_{k^2-1}^*$.*

Proof. We set initially $y^{**} := y^*$. If there is any interval $I = I_1 \dot{\cup} I_2 \in \mathcal{I}_0 \cup \dots \cup \mathcal{I}_{k^2-1}$ with $\Delta(J(I, y^{**})) \geq \delta \cdot |I|$, then we select the job $j \in J(I, y^{**})$ that (by symmetry) has $|\delta_{J(I, y^{**})}^+(j) \cup \{j\}| \geq \frac{\delta}{2} \cdot |I|$. Then we replace y^{**} by the Lasserre solution conditioned on the event “ $x_{j, I_2} = 1$ ”. Note that this means that all jobs in $\delta_{J(I, y^{**})}^+(j) \cup \{j\}$ will be removed from $J(I, y^{**})$. In fact, each such job will be moved to $J(I', y^{**})$ where $I' \subseteq I_2$ is some subinterval. The conditioning can also change the assignment of other jobs, but for each job j , the set $\text{supp}(j, y^{**})$ can only *shrink* if we condition on any event, see Lemma 3.b. Hence jobs only move from intervals to subintervals.

Since in each iteration, at least $\frac{\delta}{2} \cdot |I| \geq \frac{\delta}{2} \cdot \frac{T^*}{2^{k^2}}$ many jobs “move” and each job moves at most k^2 many times out of an interval in $\mathcal{I}_0^* \cup \dots \cup \mathcal{I}_{k^2-1}^*$, we need to condition at most

$$\frac{2mT^* \cdot k^2}{\delta \frac{T^*}{2^{k^2}}} = 2mk^2 \cdot \frac{2^{k^2}}{\delta}$$

many times. □

The implication of Lemma 7 is that the set of jobs assigned to intervals $I \in \mathcal{I}_0^* \cup \dots \cup \mathcal{I}_{k^2-1}^*$ will not contain long chains, simply because we have only few intervals and none of the jobs assigned to intervals contain long chains anymore.

Lemma 8. *After applying Lemma 7, the longest chain within jobs assigned to intervals $I \in \mathcal{I}_0^* \cup \dots \cup \mathcal{I}_{k^2-1}^*$ has length at most $k^2 \delta T^*$.*

Proof. First, let us argue how many jobs a chain can have that are all assigned to intervals of the same level ℓ . From each interval I , the chain can only include $\delta|I| = \delta \cdot \frac{T^*}{2^\ell}$ many jobs. Since $|\mathcal{I}_\ell| = 2^\ell$, the total number of jobs from level ℓ is bounded by δT^* . The claim follows from the pigeonhole principle and the fact that we have k^2 many levels in $\mathcal{I}_0^* \cup \dots \cup \mathcal{I}_{k^2-1}^*$. □

We can summarize the algorithm from Lemma 7 as follows:

BREAKING LONG CHAINS

Input: Scheduling instance with jobs J^* , a precedence order, a Lasserre solution $y^* \in \text{LAS}_{r^*}(K(T))$, an interval I^*

Output: A Lasserre solution y^{**} with maximum chain length $k^2 \delta T^*$ in $\bigcup_{\ell=0}^{k^2-1} J(\ell, y^{**})$

- (1) Make a copy $y^{**} := y^*$
- (2) WHILE $\exists I \in \bigcup_{\ell=0}^{k^2-1} \mathcal{I}_\ell^*$ WITH $\Delta(J(I, y^{**})) > \delta|I|$ DO
 - (3) Choose a job $j \in J(I, y^{**})$ with $|\delta_{J(I, y^{**})}^+(j) \cup \{j\}| \geq \delta|I|$.
 - (4) If $|\delta_{J(I, y^{**})}^+(j) \cup \{j\}| \geq \frac{\delta}{2} \cdot |I|$ THEN condition in y^{**} on $x_{j, I_2} = 1$
ELSE condition on $x_{j, I_1} = 1$.

Note that after each conditioning in step (6), the vector y^{**} will change and the set $J(I, y^{**})$ will be updated.

5 Step (4) — Scheduling top jobs

Consider the algorithm from Section 3 and the state at the end of Step 3. At this point, we have a schedule σ that schedules most of the bottom jobs. The main argument that remains to be shown is how to add in the top jobs which are owned by intervals in $\mathcal{I}_0^* \cup \dots \cup \mathcal{I}_{\ell^*-k-1}^*$.

This is done in two steps. First, we use a *matching-based* argument to show that most top jobs can be inserted in the existing schedule so that the precedence constraints with the bottom jobs are respected. In this step, we will be discarding up to $4m \cdot 2^{-k} \cdot T^*$ many jobs. More crucially, the schedule will not have satisfied precedence constraints within J_{top} . In a 2nd step, we temporarily remove the top jobs from the schedule and reinsert them with a variant of the *Earliest Deadline First (EDF)* scheduling. As we will see later in Theorem 10, this results in at most $\frac{\varepsilon}{8 \log T} \cdot T^*$ additionally discarded jobs.

5.1 A preliminary assignment of top jobs

Let us recall what we did so far. In Step 3, we applied Lemma 6 recursively on each interval $I \in \mathcal{I}_{\ell^*}$ to schedule the bottom jobs. We already argued that the resulting schedules could be combined to a schedule $\sigma : J_{\text{bottom}} / J_{\text{bottom-discarded}} \rightarrow I^*$ that respects all precedence constraints.

Let the intervals in \mathcal{I}_{ℓ^*} be called I_1, \dots, I_p , so that the time horizon T^* is partitioned into p equally sized *subintervals* with $p = 2^{\ell^*}$. After reindexing the time horizon, let us assume for the sake of a simpler notation that $I^* = \{1, \dots, T^*\}$. If we abbreviate $t_i := i \cdot \frac{T^*}{p}$ for $i \in \{0, \dots, p\}$, then the i th interval contains the time periods $I_i := \{t_{i-1} + 1, \dots, t_i\}$. Each time t has an available *capacity* of $\text{cap}(t) = m - |\sigma^{-1}(t)| \in \{0, \dots, m\}$ many machines, which is the number of machines not used by jobs in J_{bottom} . We abbreviate $\text{cap}(I_i) := \sum_{t \in I_i} \text{cap}(t)$ as the capacity of interval I_i .

The available positions of jobs in J_{top} are constrained by the scheduled times of jobs in J_{bottom} . As we had no further control over the exact position of the bottom jobs within their intervals I_i , we want to define for each job $j \in J_{\text{top}}$ a *release time* r_j and a *deadline* d_j determined by the most pessimistic outcome of how σ could have scheduled the bottom jobs. For all $j \in J_{\text{top}}$, we define

$$\begin{aligned} r_j &:= \min \{ t_i + 1 \mid \sigma(j') \leq t_i \ \forall j' \in J_{\text{bottom}} \text{ with } j' < j \} \\ d_j &:= \max \{ t_i \mid \sigma(j') \geq t_i + 1 \ \forall j' \in J_{\text{bottom}} \text{ with } j < j' \} \end{aligned} \tag{2}$$

In particular, the release time will be the first time unit of an interval I_i and the deadline will be the last time unit of an interval I_i . Let $i_r(j)$ and $i_d(j)$ be the corresponding indices, so that the release time is of the form $r_j = t_{i_r(j)-1} + 1$ and the deadline is $d_j = t_{i_d(j)}$. Then our goal is to show that most top jobs j can be scheduled somewhere in the time frame $I_{i_r(j)} \cup \dots \cup I_{i_d(j)}$. This would imply that at least all precedence constraints between bottom and top jobs are going to be satisfied.

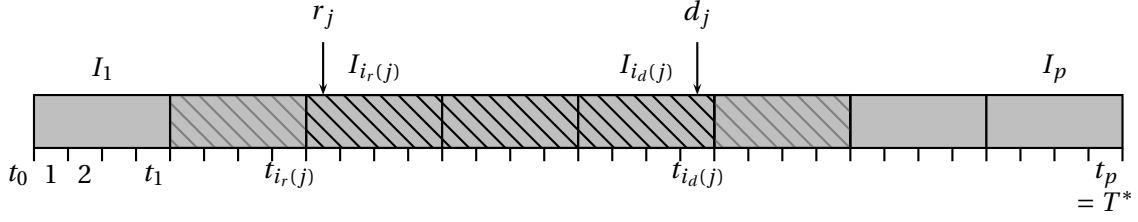


Figure 2: Visualization of interval $I^* = I_1 \dot{\cup} \dots \dot{\cup} I_p$ and possible release times and deadlines for a job $j \in J_{\text{top}}$. Note that $\text{supp}(j, y^{**})$ might be stretched over the whole hatched area, while our choice of r_j and d_j will force us to process j inside the black-hatched area (or to discard the job).

Notice here that the existing fractional assignment that y^{**} provides for $j \in J_{\text{top}}$ might be placed up to one interval outside of the range $[r_j, d_j]$ (see Figure 2). This is due to our requiring the release time and deadline to correspond to interval beginnings and endings. Let $J'_{\text{bottom}} := J_{\text{bottom}} \setminus J_{\text{bottom, discarded}}$ be the majority of bottom jobs that our schedule processes.

Lemma 9. *A valid schedule $\sigma : J'_{\text{bottom}} \rightarrow I^*$ of bottom jobs can be extended to a schedule $\sigma : J'_{\text{bottom}} \cup J'_{\text{top}} \rightarrow I^*$ with $J'_{\text{top}} \subseteq J_{\text{top}}$ that includes most of the top jobs. The schedule satisfies (i) $|\sigma^{-1}(t)| \leq m$ for $t \in I^*$; (ii) $r_j \leq \sigma_j \leq d_j$ for all $j \in J'_{\text{top}}$ and (iii) one discards at most $|J_{\text{top}} \setminus J'_{\text{top}}| \leq 4m \cdot 2^{-k} \cdot T^*$ many top jobs.*

Proof. We want to use a matching-based argument. For this sake, we consider the bipartite graph with jobs on one side and subintervals on the other. Formally, we define a graph $G = (V, U, E^+)$ with $V = J_{\text{top}}$, $U = \{1, \dots, p\}$ where the nodes $i \in U$ have capacity $\text{cap}(I_i)$, and edges

$$E^+ = \{(j, i) \in V \times U \mid \max\{i_r(j) - 1, 1\} \leq i \leq \min\{i_d(j) + 1, p\}\}.$$

Then the neighborhood of each top job includes every interval in which it is fractionally scheduled in y^{**} . Moreover, each of the bottom jobs $j \in J'_{\text{bottom}}$ has been assigned by σ to precisely that interval I_i with $y_{j, I_i}^{**} = 1$. Hence y^{**} gives a V -perfect fractional matching that respects the given capacities $\text{cap}(I_i)$. In bipartite graphs, the existence of a *fractional* V -perfect matching implies the existence of an *integral* V -perfect matching, see e.g. [Sch03].

However, in order to assign the top jobs to slots within release times and deadlines we are only allowed to use the smaller set of edges

$$E = \{(j, i) \in V \times U \mid i_r(j) \leq i \leq i_d(j)\}.$$

For any $J^* \subseteq V$, we let $N^+(J^*)$ be the neighborhood of J^* along edges in E^+ and let $N(J^*)$ be its neighborhood along edges in E . Let the magnitude of a neighborhood $|N(J^*)|$ be

defined as the sum of capacities of the nodes it contains. By *Hall's Theorem* [Sch03], the minimum number of *exposed* V -nodes in a maximum matching in E is precisely

$$\max_{J \subseteq V} \{|J| - |N(J)|\}.$$

Now, fix the set $J^* \subseteq V$ attaining the maximum; then $|J^*| - |N(J^*)|$ is the number of jobs that we have to discard. Since E^+ allows for a V -perfect matching, the reverse direction of Hall's Theorem gives that $|J^*| \leq |N^+(J^*)|$. Thus $|J^*| - |N(J^*)| \leq |N^+(J^*)| - |N(J^*)|$. Note that $N(J^*)$ is in general not a consecutive interval of $\{1, \dots, p\}$. We can upper bound the difference $|N^+(J^*)| - |N(J^*)|$ by $2m \cdot \frac{T^*}{p}$ times the number of connected components of $N(J^*)$. This is the point where we take advantage of the “gap” between the levels of the top and bottom jobs: for each job $j \in V$ there is an interval $I \in \mathcal{I}_0^* \dot{\cup} \dots \dot{\cup} \mathcal{I}_{\ell^*-k-1}^*$ so that $N^+(j)$ contains the midpoint of that interval. Due to the gap, there are at most $p \cdot 2 \cdot 2^{-k}$ such intervals². Hence the number of discarded jobs can be bounded by

$$|J^*| - |N(J^*)| \leq 2m \cdot \frac{T^*}{p} \cdot 2p \cdot 2^{-k} = 4m \cdot 2^{-k} \cdot T^*.$$

Finally note that a corresponding maximum matching can be computed in polynomial time. \square

5.2 Reassigning the top jobs via EDF

We have seen so far that we can schedule most of the bottom and top jobs so that all precedence constraints within the bottom jobs are satisfied and the top jobs are correctly scheduled between the bottom jobs that they depend on. However, the schedule as it is now ignored the precedence constraints within the top jobs. In this section, we will remove the top jobs from the schedule and then reinsert them using a variant of the *Earliest Deadline First* (EDF) scheduling policy.

For the remainder of Section 5.2, we will show a stand-alone theorem that we will use as a black box. Imagine a general setting where we have m identical machines and n jobs J , each one with integer release times r_j and deadlines d_j and a unit processing time. The EDF scheduling rule picks at any time the available job with minimal d_j for processing. It is a classical result in scheduling theory by Dertouzos [Der74] that for $m = 1$ and unit size jobs, EDF is an *optimal* policy. Here “optimal” means that if there is any schedule that finishes all jobs before their deadline, then EDF does so, too. The result extends to the case of arbitrary running times p_j if one allows *preemption*.

Now, our setting is a little bit different. For each time t , we have a certain number $\text{cap}(t) \in \{0, \dots, m\}$ of slots. Additionally, we have a precedence order that we need to respect. But we can use to our advantage that the precedence order has only short chains; moreover, the number of different release times and deadlines is limited.

Formally we will prove the following:

²It is possible that $N(j) = \emptyset$. Still $N^+(j)$ will contain a midpoint of a level $0, \dots, \ell^* - k - 1$ interval, hence we have accounted for those jobs as well. Note that such jobs would automatically get discarded.

Theorem 10. Let J be a set of jobs with release times r_j , deadlines d_j and consistent precedence constraints³ with maximum chain length C . Suppose that $\{1, \dots, T\}$ is the time horizon, partitioned into equally sized intervals I_1, \dots, I_p and all release times/deadlines correspond to beginnings/endings of those intervals. Let $\text{cap} : [T] \rightarrow \{0, \dots, m\}$ be a capacity function and assume that there exists a schedule $\tilde{\sigma} : J \rightarrow [T]$ assigning each job to slots between its release time and deadline that respects capacities but does not necessarily respect precedence constraints within J .

Then in polynomial time, one can find a schedule $\sigma : J \setminus J_{\text{discarded}} \rightarrow [T]$ that respects capacities, release times, deadlines and precedence constraints and discards $|J_{\text{discarded}}| \leq pmC$ many jobs.

We use the following algorithm, which is a variant of Earliest Deadline First, where we discard those jobs that we cannot process in time:

EARLIEST DEADLINE FIRST

Input: Jobs J with deadlines, release times, precedence constraints;

capacity function $\text{cap} : [T] \rightarrow \{0, \dots, m\}$

Output: Schedule $\sigma : J \rightarrow [T] \cup \{\text{DISCARDED}\}$

- (1) Set $\sigma(j) := \text{UNASSIGNED}$ for all $j \in J$ and $J_{\text{discarded}} := \emptyset$
- (2) Sort the jobs $J = \{1, \dots, n\}$ so that $d_1 \leq d_2 \leq \dots \leq d_n$
- (3) FOR $t = 1$ TO T DO
 - (4) FOR $\text{cap}(t)$ MANY TIMES DO
 - (5) Select the lowest index j of a job with $r_j \leq t \leq d_j$ that has not been scheduled or discarded and that has all jobs in $\delta^-(j)$ already processed (or discarded).
 - (6) Set $\sigma(j) := t$ (if there was any such job)
 - (7) FOR each $j \in J$ with $d_j = t$ and $\sigma(j) = \text{UNASSIGNED}$, add j to $J_{\text{discarded}}$ and set $\sigma(j) := \text{DISCARDED}$

At the end all jobs j will be either scheduled between r_j and d_j (that means $r_j \leq \sigma(j) \leq d_j$) or they are in $J_{\text{discarded}}$.

We will say that a job j was *discarded in the interval* $[t, t']$ if $j \in J_{\text{discarded}}$ and $t \leq d_j \leq t'$. We call a time t *busy* if $|\sigma^{-1}(t)| = \text{cap}(t)$ and *non-busy* otherwise. Let us make a useful observation:

Lemma 11. Let $I = \{t', \dots, t''\} \subseteq I_i$ be part of one of the subintervals. Suppose that there is a non-busy time $t^* \in I$ and a job j with $I \subseteq \{r_j, \dots, d_j\}$ and $\sigma(j) \in \{\text{DISCARDED}\} \cup \{t'' + 1, \dots, T\}$. Then there is a job $j^* \in \sigma^{-1}(t^*)$ with $j^* < j$.

Proof. Consider any inclusion-wise maximal chain of jobs $j_1 < j_2 < \dots < j_q$ that ends in $j = j_q$ and otherwise has only jobs $j_1, \dots, j_{q-1} \in \sigma^{-1}(\{t^*, \dots, t''\})$. First suppose that $q > 1$ and hence $j_1 \neq j$. It is impossible that $\sigma(j_1) > t^*$ because by assumption $r_{j_1} \leq t^*$

³Here “consistent” means that for a pair of dependent jobs $j < j'$ one has $r_j \leq r_{j'}$ and $d_j \leq d_{j'}$.

and hence EDF would have processed j_1 already earlier at time t^* (by maximality of the chain, there is no job scheduled at times $t^*, \dots, \sigma(j_1) - 1$ on which j_1 might depend). Hence $\sigma(j_1) = t^*$ and by transitivity $j_1 < j$, which gives the claim.

In the 2nd case, we have $j_1 = j$, hence there is no job that j depends on scheduled between t^* and t'' . But we know that $r_j \leq t^*$. Thus EDF would have processed j at time t^* or earlier. \square

With this observation we can easily limit the number of non-busy times per interval:

Lemma 12. *Let $I = \{t', \dots, t''\} \subseteq I_i$ be part of one of the subintervals. Suppose that there is at least one job j with $I \subseteq \{r_j, \dots, d_j\}$ and $\sigma(j) \in \{\text{DISCARDED}\} \cup \{t'' + 1, \dots, T\}$. Then the number of non-busy times in I is bounded by C .*

Proof. By Lemma 11, for the latest time $t^* \in I$ with $|\sigma(t^*)| < \text{cap}(t^*)$, there is at least one job $j^* \in \sigma^{-1}(t^*)$ with $j^* < j$. Then we can continue by induction, replacing t'' with $t^* - 1$ and replacing j by j^* to build a chain of jobs ending with j that includes a job scheduled at each non-busy time. Since no chains can be longer than C , this gives the claim. \square

Now we come to the main argument where we give an upper bound of the number of discarded jobs:

Lemma 13. *One has $|J_{\text{discarded}}| \leq pmC$.*

Proof. Suppose that $|J_{\text{discarded}}| \geq p \cdot K$; we will then derive a bound on K . By the pigeon-hole principle, we can find an interval I_b so that at least K many jobs get discarded in I_b . Let us denote the lowest priority (i.e. the latest deadline) job that gets discarded in I_b by j_s . Now delete all those lower priority jobs j_{s+1}, \dots, j_n . Note that this does not affect how EDF schedules j_1, \dots, j_s and still we would have at least K jobs discarded in I_b , including j_s . By Lemma 12, the number of non-busy periods in I_b is bounded by C . Now, choose a minimal index $a \in \{1, \dots, b-1\}$ so that in each of the intervals I_a, \dots, I_b one has at most C many non-busy periods. We abbreviate $I' := I_a \cup \dots \cup I_b$. Note that by definition I_{a-1} has more than C many non-busy periods⁴. Define

$$J' := \{j \in \{j_1, \dots, j_s\} : (\sigma(j) \in I') \text{ or } (j \text{ discarded and } d_j \in I')\}.$$

By Lemma 11, any job in J' has its release time in I_a or later, since otherwise we could not have $C+1$ non-busy times in I_{a-1} . Now, let us double count the number of jobs in J' . On the one hand, we have

$$\begin{aligned} |J'| &\geq \sum_{i=a}^b |\sigma^{-1}(I_i)| + |J' \cap J_{\text{discarded}}| \geq \sum_{i=a}^b (\text{cap}(I_i) - mC) + \underbrace{|J' \cap J_{\text{discarded}}|}_{\geq K} \\ &\geq \text{cap}(I') - pmC + K. \end{aligned}$$

⁴Admittedly it is possible that $a = 1$ in which case one might imagine I_0 as an interval in which all times are non-busy and which does not contain any release times.

On the other hand, we know that there is an assignment $\tilde{\sigma}$ of jobs in J' to slots in I_a, \dots, I_b . That tells us that $|J'| \leq \text{cap}(I')$. Comparing both bounds gives that $K \leq pmC$. \square

6 Step (2) — Accounting for discarded jobs

In this section, we need to argue that the level ℓ^* can be chosen so that the total number of jobs that are discarded in Steps (1)-(4) are bounded by

$$|J_{\text{discarded}}^*| \leq \frac{\varepsilon}{2} \cdot \frac{\log_2(|I^*|)}{\log_2(T)} \cdot |I^*| + \frac{\varepsilon}{2m} \cdot |J^*|$$

as claimed. Let us summarize the three occasions in the algorithm where a job might get discarded:

- (A) In Step (3), in order to schedule the bottom jobs, we have 2^{ℓ^*} many recursive calls of Lemma 6 for intervals $I \in \mathcal{I}_{\ell^*}^*$. The cumulative number of discarded jobs from all those calls is bounded by

$$2^{\ell^*} \cdot \frac{\varepsilon}{2} \cdot \frac{\log_2(\frac{T^*}{2^{\ell^*}})}{\log_2(T)} \cdot \frac{T^*}{2^{\ell^*}} + \frac{\varepsilon}{2m} \cdot |J_{\text{bottom}}| = \frac{\varepsilon}{2} \cdot \frac{\log_2(T^*) - \ell^*}{\log_2(T)} \cdot T^* + \frac{\varepsilon}{2m} \cdot |J_{\text{bottom}}|$$

- (B) As we have seen in Section 5, the number of top jobs that need to be discarded in Step (4) can be bounded by

$$4m \cdot 2^{-k} \cdot T^* + pmC \leq 4m \cdot 2^{-k} \cdot T^* + k^2 m 2^{2k^2} T^* \delta \leq \frac{\varepsilon}{4} \cdot \frac{1}{\log_2(T)} \cdot T^*.$$

Here we use that the length of the maximum chain within the top jobs is $C \leq k^2 \delta T^*$. Moreover, we have substituted the parameters $p = 2^{\ell^*} \leq 2^{k^2}$ as well as $\delta = \frac{\varepsilon}{8k^2 m 2^{2k^2} \log_2(T)}$ and $k = c_1 \frac{m}{\varepsilon} \log_2 \log_2(T)$ with a large enough constant $c_1 > 0$.

- (C) In Step (2), we discard all the middle jobs. In the remainder of this section we prove that there is an index ℓ^* so that

$$|J_{\text{middle}}| \leq \frac{\varepsilon}{4} \cdot \frac{1}{\log_2(T)} \cdot T^* + \frac{\varepsilon}{2m} \cdot (|J_{\text{middle}}| + |J_{\text{top}}|)$$

Let us first assume that we can indeed find a proper index ℓ^* so that the bound in (C) is justified. Then the total number of jobs that the algorithm discards is

$$\begin{aligned} & \overbrace{\frac{\varepsilon}{2} \cdot \frac{\log_2(T^*) - \ell^*}{\log_2(T)} \cdot T^* + \frac{\varepsilon}{2m} \cdot |J_{\text{bottom}}|}^{(A)} + \overbrace{\frac{\varepsilon}{4} \cdot \frac{1}{\log_2(T)} \cdot T^*}^{(B)} + \overbrace{\frac{\varepsilon}{4} \cdot \frac{1}{\log_2(T)} \cdot T^* + \frac{\varepsilon}{2m} \cdot (|J_{\text{middle}}| + |J_{\text{top}}|)}^{(C)} \\ & \leq \frac{\varepsilon}{2} \cdot \frac{\log_2(T^*)}{\log_2(T)} + \frac{\varepsilon}{2m} \cdot \underbrace{(|J_{\text{top}}| + |J_{\text{middle}}| + |J_{\text{bottom}}|)}_{=|J^*|} \end{aligned}$$

which is the bound that we claimed in Lemma 6.

It remains to justify the claim in (C). We abbreviate $\alpha_i := |J(i \cdot k, y^{**}) \cup \dots \cup J((i+1) \cdot k - 1, y^{**})|$ for $i \in \{0, \dots, k-1\}$. In words, each number α_i represents the number of jobs owned by k consecutive levels. We observe that if there is an index $i \in \{0, \dots, k-1\}$ so that

$$(I) \quad \alpha_i \leq \frac{\varepsilon}{4 \log_2(T)} \cdot T^* \quad \text{or} \quad (II) \quad \alpha_i \leq \frac{\varepsilon}{2m} \cdot \sum_{j=1}^i \alpha_j$$

then we can choose $\ell^* := (i+1) \cdot k$ and (C) will be satisfied. Here we use that for this particular choice of ℓ^* , we will have $|J_{\text{middle}}| = \alpha_i$ and $|J_{\text{top}}| = \alpha_0 + \dots + \alpha_{i-1}$.

So, we assume for the sake of contradiction that no index i satisfies either (I) or (II) (or both). Then one can easily show that the α_i 's have to grow exponentially. We show this in a small lemma:

Lemma 14. *Let $q \in \mathbb{N}$ and suppose we have a sequence of numbers $\alpha_0, \alpha_1, \dots, \alpha_N$ satisfying $\alpha_i \geq \alpha_{\min} > 0$ and $\alpha_i \geq \frac{1}{q} \cdot \sum_{j=1}^i \alpha_j$ for all $i = 0, \dots, N$. Then $\alpha_i \geq 2^{\lfloor i/(2q) \rfloor} \cdot \alpha_{\min}$.*

Proof. Group the indices into consecutive blocks of $2q$ numbers, where $\alpha_0, \dots, \alpha_{2q-1}$ is block 0, $\alpha_{2q}, \dots, \alpha_{4q-1}$ is block 1 and so on. We want to prove by induction that each α_i in the j th block is at least $2^j \cdot \alpha_{\min}$. For $j = 0$, the claim follows from the assumption. For $j > 0$, we use that α_i is at least the sum of $2q$ numbers that by inductive hypothesis are all at least $2^{j-1} \cdot \alpha_{\min}$. The claim follows. \square

Applying Lemma 14 with $\alpha_{\min} := \frac{\varepsilon}{4 \log_2(T)} \cdot T^*$ and $q := \frac{2m}{\varepsilon}$ we obtain in particular that

$$\alpha_{k-1} \geq 2^{\lfloor (k-1) \frac{\varepsilon}{4m} \rfloor} \cdot \frac{\varepsilon}{4 \log_2(T)} \cdot T^*$$

If we set $k = \frac{c_1 m}{\varepsilon} \log_2 \log_2 T$ for some adequately large c_1 , then $\alpha_{k-1} > m T^*$, which is a contradiction since we only have $|J^*| \leq m \cdot |I^*| = m T^*$ many jobs with $y_{j,I^*} = 1$.

7 Conclusion

For the proof of Lemma 6 we already argued that the number of discarded jobs satisfies the claimed bound and that all precedence constraints will be satisfied. What concerns the number of rounds in the hierarchy, recall that we started with a vector $y^* \in \text{LAS}_{r^*}(K(T))$ with $r^* \geq \log_2(T^*) \cdot 2mk^2 \cdot 2^{k^2}/\delta$. Then we apply a round of conditionings in Lemma 7 to obtain $y^{**} \in \text{LAS}_{r^{**}}(K(T))$ with $r^{**} := r^* - 2mk^2 \cdot 2^{k^2}/\delta$. We use copies of the vector y^{**} in our recursive application of Lemma 6 to intervals of size $T^*/2^{\ell^*}$. Since $\ell^* \geq k \geq 1$, the remaining number of Lasserre rounds satisfies $r^{**} \geq \log_2(T^*/\ell^*) \cdot 2mk^2 \cdot 2^{k^2}/\delta$. Thus we still have enough Lasserre rounds for the recursion.

Another remark concerns why we may assume that the precedence constraints are consistent in Theorem 10. Suppose we have jobs $j, j' \in J_{\text{top}}$ with $j < j'$ and consider the definition of release times and deadlines in Eq. 2. By transitivity, any job $j'' \in J_{\text{bottom}}$

with $j' < j''$ which limits the deadline of j' will also limit the deadline of j , hence $d_j \leq d_{j'}$. Similarly one can argue that $r_j \leq r_{j'}$.

This concludes the proof of Lemma 6 and our main result follows.

References

- [ABS10] S. Arora, B. Barak, and D. Steurer. Subexponential algorithms for unique games and related problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 563–572, 2010.
- [ARV09] S. Arora, S. Rao, and U. V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.
- [BK09] N. Bansal and S. Khot. Optimal long code test with one free bit. In *Foundations of Computer Science, 2009. FOCS '09. 50th Annual IEEE Symposium on*, pages 453–462, Oct 2009.
- [Cha95] M. Charikar. Approximation algorithms for problems in combinatorial optimization. Technical report, B. Tech. Project Report, Department of Computer Sc. and Engg., IIT Bombay, 1995.
- [Chl07] E. Chlamtac. Approximation algorithms using hierarchies of semidefinite programming relaxations. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 691–701, 2007.
- [CKR10] E. Chlamtac, R. Krauthgamer, and P. Raghavendra. Approximating sparsest cut in graphs of bounded treewidth. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, pages 124–137, 2010.
- [CS08] E. Chlamtac and G. Singh. Improved approximation guarantees through higher levels of SDP hierarchies. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, pages 49–62, 2008.
- [Dag10] Open problems – scheduling. In Susanne Albers, Sanjoy K. Baruah, Rolf H. Möhring, and Kirk Pruhs, editors, *Scheduling*, number 10071 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2010. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

- [Der74] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *IFIP Congress*, pages 807–813, 1974.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, New York, 1979.
- [GR08] D. Gangal and A. Ranade. Precedence constrained scheduling in optimal. *Journal of Computer and System Sciences*, 74(7):1139 – 1146, 2008.
- [Gra66] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.
- [GTW13] A. Gupta, K. Talwar, and D. Witmer. Sparsest cut on bounded treewidth graphs: algorithms and hardness results. In *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*, pages 281–290, 2013.
- [GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.
- [Las01a] J. Lasserre. An explicit exact SDP relaxation for nonlinear 0-1 programs. In *IPCO*, pages 293–303, 2001.
- [Las01b] J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.
- [Lau03] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. *Math. Oper. Res.*, 28(3):470–496, 2003.
- [LK78] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978.
- [LS77] S. Lam and R. Sethi. Worst case analysis of two scheduling algorithms. *SIAM J. Comput.*, 6(3):518–536, 1977.
- [Rot13] T. Rothvoss. Mapsp tutorial: The lasserre hierarchy in approximation algorithms, 2013.
- [RT12] P. Raghavendra and N. Tan. Approximating csps with global cardinality constraints using SDP hierarchies. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 373–387, 2012.
- [Sch03] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

- [Sve10] O. Svensson. Conditional hardness of precedence constrained scheduling on identical machines. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 745–754, New York, NY, USA, 2010. ACM.
- [SW99] P. Schuurman and G. J. Woeginger. Polynomial time approximation algorithms for machine scheduling: ten open problems. *Journal of Scheduling*, 2(5):203–213, 1999.